# Documentazione sistema Chat Server-Client

**1. Scopo dell'elaborato:** la realizzazione di un sistema di chat implementato attraverso un server multithread per gestire connessioni asincrone e un client in grado di connettersi al server e sfruttare le sue funzionalità di chat. Vengono utilizzati per questo scopo i socket, oggetti che permettono l'invio e la ricezione di dati tra host remoti. Il server e' in grado di gestire più client contemporaneamente e consente agli utenti (client) di inviare e ricevere messaggi in una chatroom condivisa.

**2. Descrizione del sistema:** il sistema è composto da server e client. Il server (chat_server.py) e il client (chat_client.py). Il server si occupa dell'accettazione, della gestione dei dati inviati e ricevuti dai client connessi e della disconnessione dei client stessi, sia volontaria che per errore. Il client offre un'interfaccia utente interattiva per l'invio e la ricezione di messaggi in modo da facilitare il processo per l'utente e dunque si limita ad operare come interfaccia verso il server.

**3. Server:** il server implementa le seguenti funzionalità:

- **Creazione del server:** attraverso un indirizzo e una porta (che devono essere validi, in caso contrario è presente un sistema per provare a creare un server con indirizzo e porta predefiniti localhost-53000).
- **Accettazione connessioni dei client:** il server accetta il tentativo di connessione del client e avvia un thread separato per ciascun di essi per gestire le loro interazioni. In questo modo viene garantita la loro gestione contemporanea.
- **Gestione comunicazione tra client:** il server permette a tutti i client connessi di inviare messaggi che vengono visualizzati in maniera globale.
- **Gestione disconnessione dei client:** il server gestisce la disconnessione, sia volontaria che non, dei client e avvisa gli altri client connessi dell'avvenuto.

**4. Client:** il client implementa le seguenti funzionalità:

- **Connessione al server:** il client prova a connettersi al server utilizzando l'indirizzo e la porta forniti.
- **Invio e ricezione dei messaggi:** l'utente può digitare un messaggio e premere il pulsante invio (sulla GUI, o alternativamente da tastiera) per inviare il messaggio al server che poi lo diffonde a tutti gli altri componenti della chat room. Il client rimane inoltre in ascolto in modo da ricevere i dati inviati dal server (che siano messaggi da parte di altri client o avvisi di altro genere su ciò che sta succedendo).
- **Disconnessione e chiusura della GUI:** Il client si occupa della chiusura del socket e della finestra d'interfaccia (GUI), in caso l'utente digiti {quit} e prema invio, prema il pulsante rosso in alto a destra per chiudere la finestra o il server venga chiuso in maniera improvvisa con il client ancora connesso.

**5. Requisiti per l'esecuzione del codice:** per eseguire correttamente il sistema, è necessario avere:

- Hardware: computer dotato di connessione locale o internet a seconda degli host che utilizzano il sistema di chat.
- Software: (l'ultima versione di) Python installato. Vengono inoltre utilizzate librerie come socket e threading e la libreria Tkinter (per la GUI da parte client).

**6. Come eseguire il codice:** è necessario eseguire gli script forniti chat_server.py e chat_client.py senza un ordine preciso in particolare. Successivamente è importante stando attenti all'ordine, avviare il server inserendo l'indirizzo e la porta che si vogliono utilizzare e connettersi tramite il client specificando lo stesso indirizzo e porta (l'inserimento dei dati viene effettuato da terminale). Inserire indirizzi e porte non validi provocherà effetti diversi descritti nel punto seguente.

**7. Gestione degli errori e delle eccezioni:** il sistema è realizzato in modo da gestire correttamente ove possibile errori sia umani che non. Ad esempio, inserire dati non validi quando viene richiesto un indirizzo e una porta per avviare il server, causerà un errore, avvisando l'utente e provando ad avviare il server con dati predefiniti (localhost-53000). Un ulteriore fallimento provocherà la chiusura del programma avvisando l'utente che non è possibile aprire il server nemmeno con i dati predefiniti e fornendo informazioni riguardo alle eccezioni scaturite. Da lato client, fornire dati erronei o sollevare un'eccezione durante il tentativo di connessione causerà direttamente la chiusura del programma e l'avviso che non e' possibile connettersi al server. Viene poi in maniera prolissa effettuata una gestione delle eccezioni durante la comunicazione tra client e la loro disconnessione volontaria o non. Viene inoltre gestito anche il caso in cui sia il server a chiudere improvvisamente mentre sono presenti client connessi.